// \*\*\* Author: Group B264C \*\*\*

// \*\*\* Project: Miniproject \*\*\*

// \*\*\* Node: Spawn node \*\*\*

// \*\*\* Created: 31-12-2015 \*\*\*

#include <ros/ros.h> // Declarations

#include <turtlesim/Spawn.h>

#include <turtlesim/Pose.h>

#include <sstream>

using namespace std;

int main (int argc, char \*\*argv) {

// Initializing the ROS system and registering our program as a node

ros::init (argc, argv, "spawn\_turtle");

ros::NodeHandle nh;

// Creating and declaring a publisher object,  
 // and making it publish on a topic called "/Miniproject/Customer"

ros::Publisher posPub = nh.advertise<turtlesim::Pose>

("/Miniproject/Customer", 1000);

// Creating a rate object and setting it to 10 messages per second

ros::Rate rate(10);

// Creating and declaring a client object for the spawn service

ros::ServiceClient spawnClient = nh.serviceClient<turtlesim::Spawn>("spawn");

// Creating request and response objects

turtlesim::Spawn::Request req;

turtlesim::Spawn::Response resp;

cout<<"\n\n\n \*\*\*\*\* TURTLE SPAWN NODE \*\*\*\*\* \n\n";

// Giving values to the fields of request object

cout<<"Where do you want the turtle to spawn?"<<endl;

cout<<"Enter x coordinate between 0 and 11: ";

cin >> req.x;

while (req.x<0 || req.x>11){

cout<< "Please enter a number between 0 and 11 : "<<endl;

cin >>req.x;

}

cout<<"\n Enter y coordinate between 0 and 11: ";

cin >> req.y;

while (req.y<0 || req.y>11){

cout<< "Please enter a number between 0 and 11 : "<<endl;

cin >>req.y;

}

// We needed to invert the y axis and add 11.5,

// because for some reason the "spawn" service switches it

req.y = -req.y + 11;

req.theta = M\_PI/2;

req.name = "customer";

// We create a Pose object and give values to its fields

turtlesim::Pose spawnPos;

spawnPos.x = req.x;

spawnPos.y = -req.y + 11;

spawnPos.theta = req.theta;

// Calling the service

bool success = spawnClient.call(req, resp);

if (success){

ROS\_INFO\_STREAM("Spawned a turtle named: " << resp.name);

posPub.publish(spawnPos); // Publishing the Pose of our turtle

rate.sleep();

ros::spin();

}

else{

ROS\_ERROR\_STREAM("Failed to spawn.");

}

}